第十八章　软件架构解耦

王璐璐 wanglulu@seu.edu.cn
廖力　　lliao@seu.edu.cn

第18章　软件架构解耦

- 引言
- 微内核架构及其解耦方案
- 微服务架构及其解耦方案
- 本章小结

18.1 引言

- 耦合度表示模块之间（例如，类与类之间、子程序与子程序之间）关系的紧密程度，“低耦合度”是软件设计的目标。

- 低耦合模块（类或子程序）之间的关系尽可能简单，彼此之间的相互依赖性小，也就是“松散耦合”。

18.1 引言

- 解耦（decouple）就是降低模块之间的耦合度，也就是尽可能使得模块之间的耦合是松散耦合。

- 解耦能够保持组件之间的自主和独立性。它的直接结果就是改动成本低，维护成本低，可读性高。

18.1 引言

- 微观角度解耦
  - 选择合理的设计模式进行面向对象程序设计的解耦。
  - 例如：适配器模式（Adapter）将一个类的接口转换成用户希望得到的另一种接口。它使原本不相容的接口得以协同工作。
  - 再如：桥接模式（Bridge）将类的抽象部分和它的实现部分分离开来，使它们可以独立地变化。

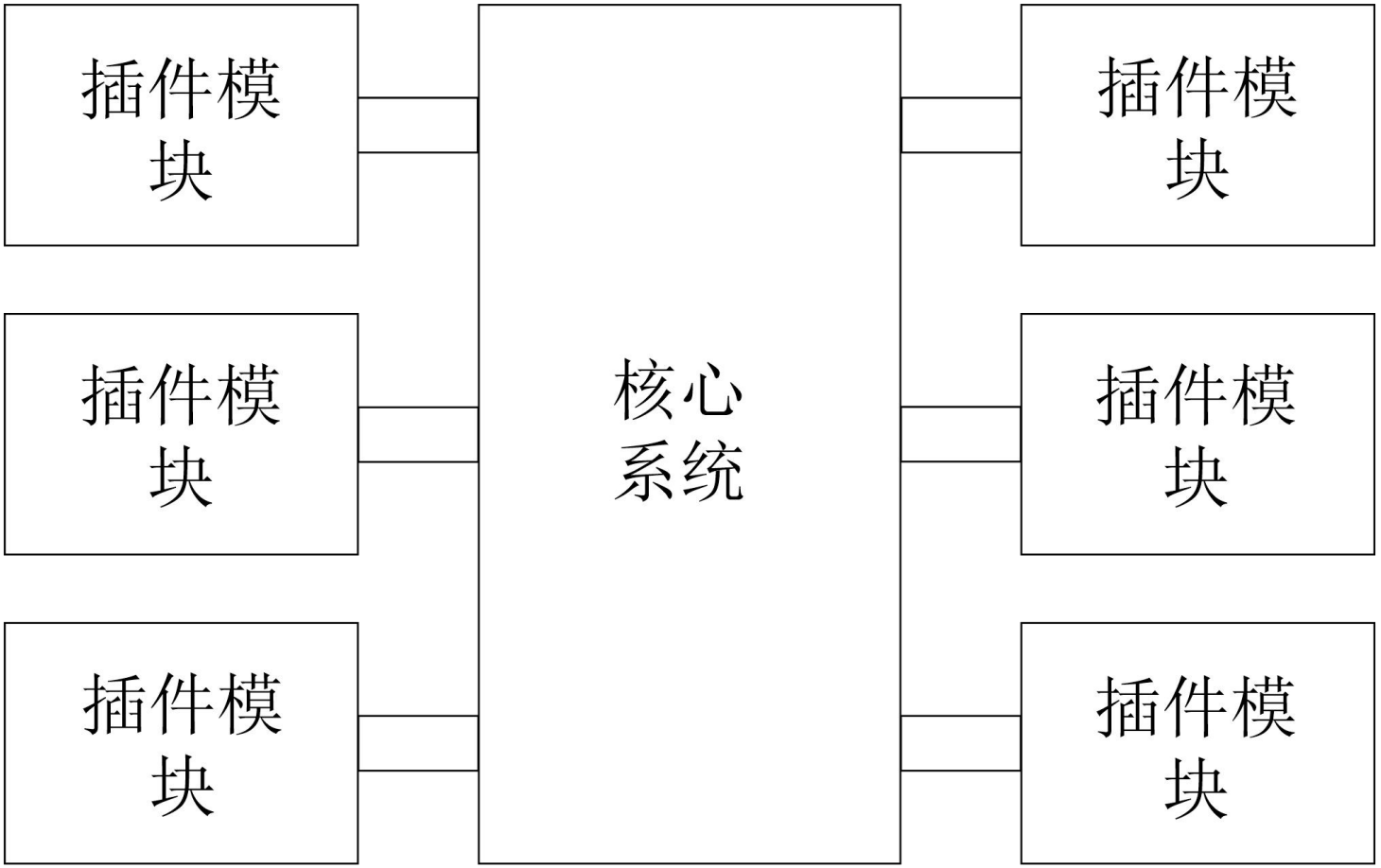
18.1 引言

- 宏观角度解耦
  - 一个结构混乱、系统组件没有内聚、掺杂大量没有必要的耦合、松弛而模糊的软件架构，将导致每个代码组件编写得不好，还会导致重复的代码和工作。
  - 从架构层面，基于架构风格，从架构风格（或模式）本身的角度去分析其解耦。

18.3微内核架构及其解耦应用

- 18.3.1模式描述与解耦
  - 微内核架构（Microkernel architecture）模式也被称为插件架构（plugin architecture）模式。
  - 应用逻辑被分割为独立的插件模块和核心系统，可以提供可扩展的，灵活的，特性隔离的功能。
  - 微内核包含两个组件：
    - 核心系统（core system）
    - 插件模块（plug-in modules）。

18.3微内核架构及其解耦应用



18.8 微内核架构结构

18.3微内核架构及其解耦应用

- 例如：美国保险公司索赔处理的改造
  - 索赔过程复杂，每个阶段都有很多不同的规则和条例来说明是否应该得到赔偿。
    - 例如汽车挡风玻璃被岩石击碎，有的州是允许赔偿的，有的是不允许的。
  - 通常保险索赔应用都会使用一个大型的复杂的规则引擎来处理。
  - 但是规则引擎会像滚雪球一样越来越大，修改一个规则可能会影响其它的规则，这种紧耦合的应用设计会造成很多问题。

18.3微内核架构及其解耦应用

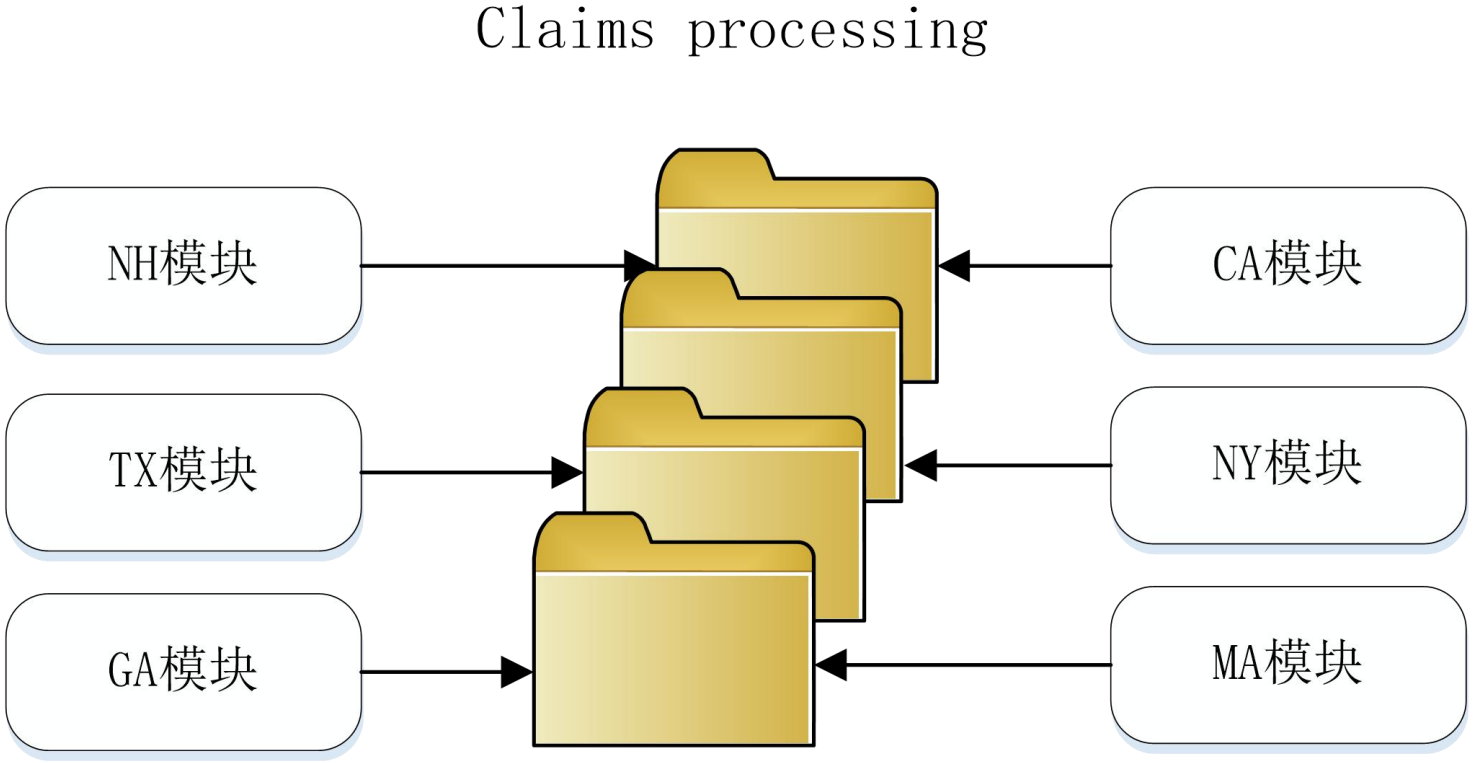
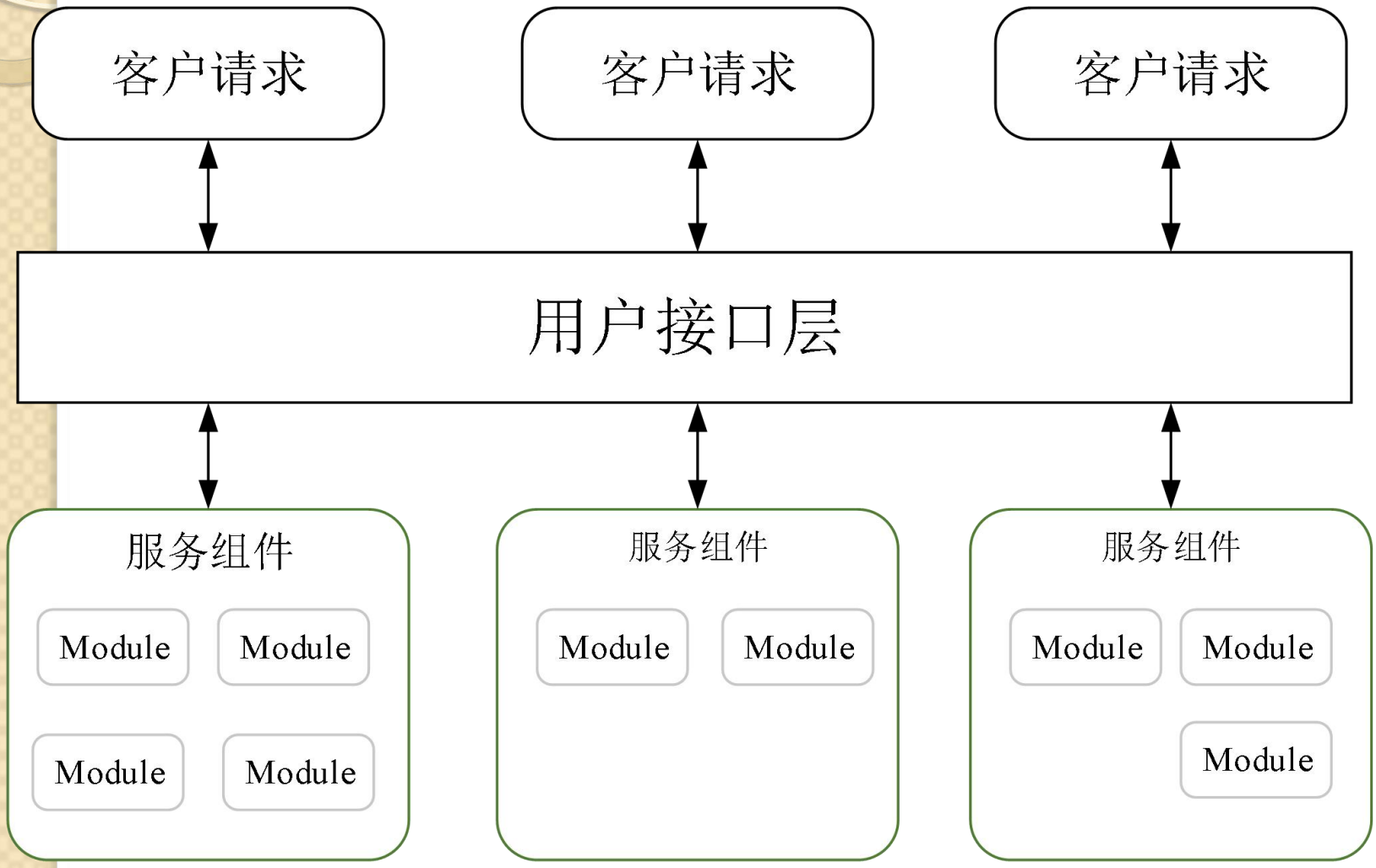


图18.9 微内核架构案例

18.4微服务架构及其解耦应用

- 18.4.1模式描述与解耦
- 微服务架构（Microservice Architect）是一种将单个应用程序开发为一组小型服务的方法，每个小型服务都在自己的进程中运行，并与轻量级机制（通常是HTTP资源API）进行通信。
- 这些服务围绕业务功能构建，可通过全自动部署机制独立部署。
- 这些服务的集中式管理的最小限度是最小的，其可以用不同的编程语言编写并且使用不同的数据存储技术。
- 提倡将单块架构的应用划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。

18.4微服务架构及其解耦应用



18.4微服务架构及其解耦应用

- 例如：一个合同管理系统的改造
  - 该系统基于.NET，SAGE CRM二次开发的产品。
  - 系统架构过于陈旧，性能、可靠性无法满足现有的需求，而且功能繁杂、结构混乱，定制的代码与SAGE CRM系统耦合度极高。

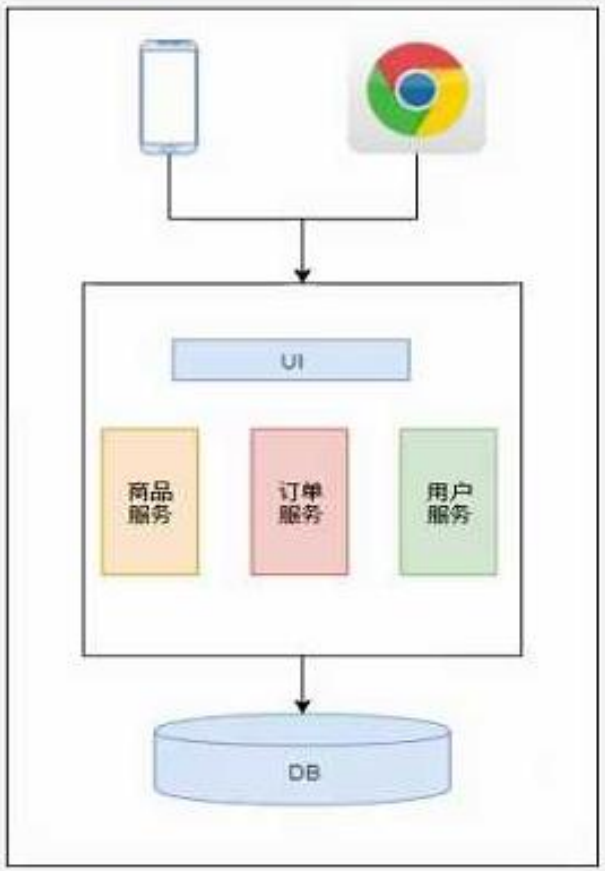
18.4微服务架构及其解耦应用

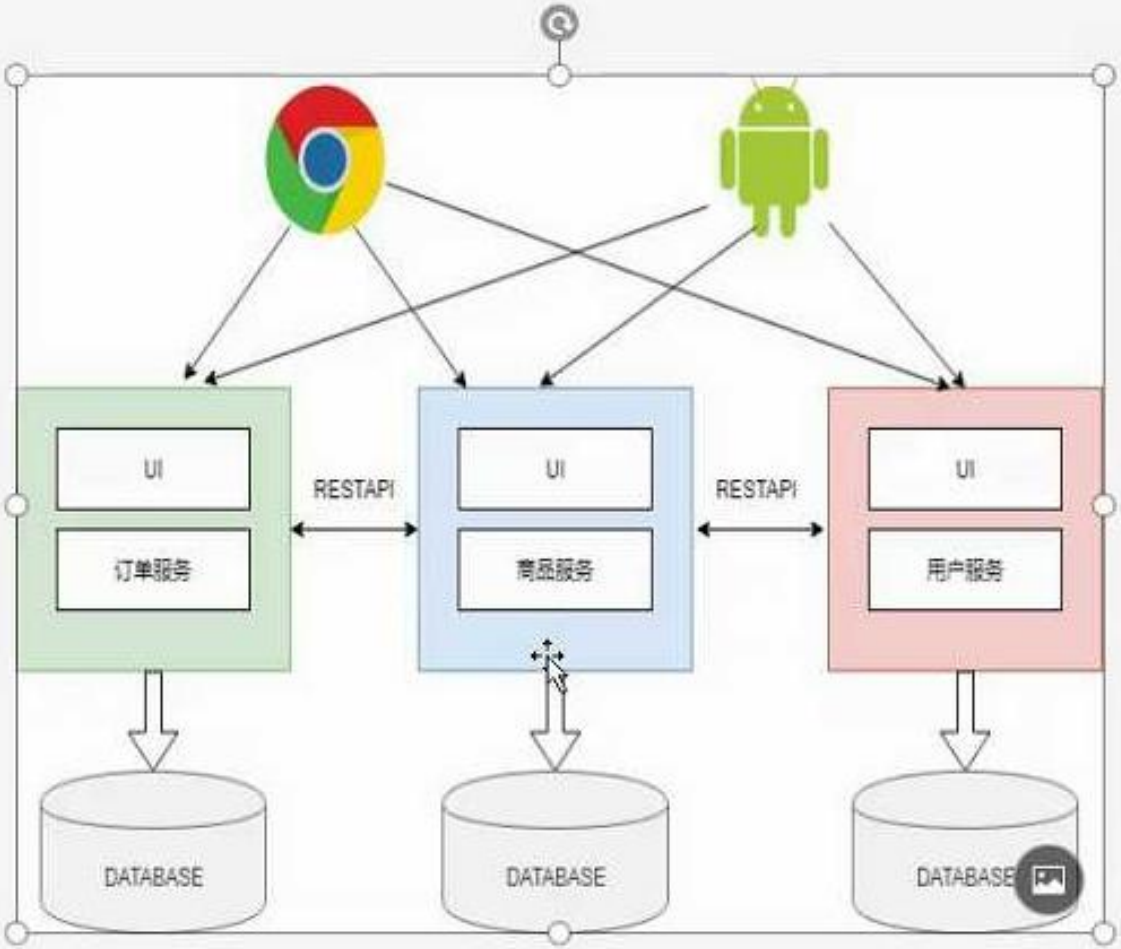
- 改造过程
  - 秉持最小修改原则；
  - 功能剥离：在现有合同管理系统的外围，逐步构建功能服务接口，将系统核心的功能分离出来；同时，使用代理机制，将用户对原有系统的访问，转发到新的服务中，从而解耦原合同系统与用户之间的依赖。
  - 数据解耦：随着部分功能的解耦，已经存在一些微服务能够独立为用户提供功能。这时，逐渐考虑将数据解耦，从原有的单块架构数据中剥离相关的业务数据。尽量满足对于每个服务，有独立的、隔离的业务数据系统。

18.4微服务架构及其解耦应用

- 改造过程
  - 数据同步：对于复杂的业务逻辑，在新的服务（业务逻辑、数据）独立出来后，可采用ETL的机制，将服务中的业务数据同步到单块架构的数据库中，保障原有的功能能够被继续使用。
    - ETL是将业务系统的数据经过抽取（Extract）、清洗转换（Transform）之后加载（Load）到数据仓库的过程，目的是将企业中的分散、零乱、标准不统一的数据整合到一起，为企业的决策提供分析依据。
  - 通过如上述方式不断迭代，逐步将功能解耦成独立的服务（包括业务逻辑以及业务数据等）。

18.4微服务架构及其解耦应用





本章小结

- 本章详细地讨论了两个案例，说明了其软件架构耦合度增加的原因，通过两种典型的软件架构风格的特征分析，讨论了对这两个案例软件架构进行解耦的基本思想。
- 由于软件架构设计和演化过程中，组件（连接件）之间的低耦合是主要的原则之一，遵守设计原则、提高软件架构自身的质量，是软件架构设计始终最求的目标，所以解耦在软件架构生命周期中是不可或缺的活动。
- 然而，目前软件架构的有效解耦方法并不是很多，采用的技术手段还存在很大不足，需要相关从业人员努力为之，发现更好的软件架构解耦技术。

Thanks